# club commodore

Boletín informativo para los usuarios de microordenadores

**VIC y CBM**







Nº/1

octubre 1982

# MICROELECTRÓNICA Y CONTROL, S.A. en SONIMAG-20

Desde el Salón de la Imagen, el Sonido y la Electrónica, SONIMAG, que se ha celebrado en Barcelona del 27 de septiembre al 3 de octubre, tenemos el placer de mandar un saludo a los lectores de CLUB COMMODORE. En esta página pueden verse varios aspectos del stand de MICROELECTRÓNICA Y CONTROL, S.A. que muestran la expectación existente en torno al VIC-20. Como noticia fresca adelantamos que, a partir del número 3 (diciembre), CLUB COMMODORE va a aparecer con DIECISÉIS PÁGINAS, DIECISÉIS (¡el doble!). En el mes de noviembre estaremos en el SIMO en Madrid con el VIC-20 y más novedades. Esperamos tener la misma acogida que en el certamen de Barcelona aunque os rogamos que no os apelotonéis, pues parte del personal está medio asfixiado y con los nervios hechos un flan. ¡Hasta pronto!












## MAPA DE MEMORIA DEL VIC

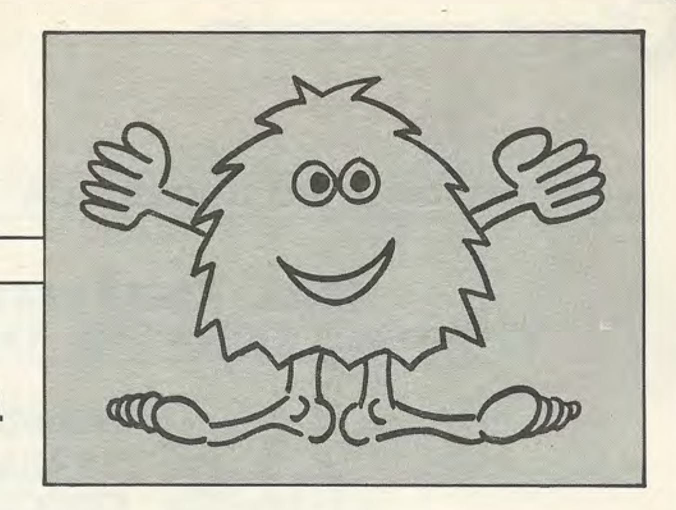
En este número se inicia una sección que esperamos sea muy útil. Se trata de ir dando en forma de coleccionable (siguiendo la tradición de tantas obras en fascículos de las que —como del cerdo— se aprovecha todo) una larga serie de notas técnicas sobre los ordenadores COMMODORE y los circuitos de MOS TECHNOLOGY que los componen. Estas notas pretenden ser una referencia para complementar los artículos de la Revista y los manuales que acompañan a los equipos. Dada su naturaleza poco literaria y para minimizar las incursiones de nuestro amigo BUG se darán en versión original —en inglés— aunque en las referencias que se hagan en los artículos se darán las explicaciones pertinentes. En realidad, el problema del idioma no es tan grave como le puede parecer al que se inicia, dado el bajo contenido de lenguaje de estas hojas. Por el contrario, su capacidad informativa está altamente condensada y, en consecuencia, son objetivamente didácticas.

VER CONTRAPORTADA POSTERIOR

Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona

## EDITORIAL

# presentación de la mascota del club commodore



*En este número 1 de CLUB COMMODORE vamos a presentar a un personaje que nos va a ayudar a amenizar las páginas de la revista. Su nombre es BUG y es ese personaje tan misterioso el que tiene la culpa de que un programa que acabamos de entrar, y cuya primera línea es la 10, al ponerlo en marcha nos responda "? SYNTAX ERROR IN 10 —o algo peor— sin ninguna consideración hacia nuestros esfuerzos en entrar el programa sin errores. Otra de sus habilidades consiste en hacer que un programa perfecto y, que después de no pocos esfuerzos, ya funciona sin errores de sintaxis nos diga que la raíz cuadrada de 2 es 9. También es el responsable de los errores pasados, presentes y futuros que puedan aparecer en esta revista (como puede verse, nos curamos en salud). Así pues ya tenemos a BUG para que cargue con todas las culpas sin propietario que haya por ahí (más de un ministro pagaría por una cosa así). Aunque, en el fondo, es un bicho simpático y buenazo que con sus pícaras actividades nos ayuda a pulir y a depurar nuestra técnica de programación, y no suele molestarase si, en un momento de ofuscación, le mentamos los progenitores.* ■

## mini-notas

**EL CURSOR**

**El cursor es el cuadrado intermitente que nos señala donde vamos a escribir el siguiente carácter y en este sentido trabaja como el indicador que hay en cualquier máquina de escribir. Cuando un programa está trabajando, el VIC está ocupado y el cursor no aparece porque no se puede aceptar información, sólo reaparece cuando se necesita entrar algún valor desde el teclado y cuando acaba la ejecución del programa con lo que aparece la palabra READY.**

**LA TECLA RETURN**

**Puede decirse que la tecla RETURN es la más importante del teclado. Cumple dos funciones: la primera es similar al retorno del carro en una máquina de escribir, es decir, volver a empezar una línea; la otra es la de decir al VIC cuándo hemos terminado de entrar un dato y que puede tomar el control del programa. Hay tres casos diferentes en que esto ocurre: cuando se ha entrado una orden en modo directo (NEW, LIST, RUN o cualquier otro), cuando hemos terminado de escribir una línea y deseamos empezar otra y en tercer lugar hacerle saber al VIC que hemos terminado de entrar un dato en una instrucción INPUT.**

**P. M.**

## VIC-20

# ALGUNOS "POKES" INTERESANTES

**El acceso a las variables internas puede resultar muy conveniente para que nuestros programas tengan un "mejor acabado"**

por **P. MASATS**

Hemos recopilado en este artículo unos cuantos «POKES» que pueden convertirse en el detalle que saque a nuestros programas del «montón» y de paso los haga más eficaces. Los hemos extraído de algunas informaciones que hemos ido interceptando con la audacia e intrepidez que la atención de nuestros lectores nos exige (¡anda ya...!)

Si deseamos realizar la misma función de CTRL-color con un POKE:

POKE 646,X donde X es el número que resulta de restarle uno a la cifra de la tecla del color del VIC (es decir, para obtener la impresión en púrpura hay que hacer: POKE 646,4).

Para suspender la función del teclado:

POKE 649,0 y POKE 649,10 «resucita» el teclado.

Si hacemos POKE 650,255 permitimos que todas las teclas tengan repetición. Con POKE 650,67 ninguna tecla repite si se mantiene pulsada. Al hacer POKE 650,0 volvemos a las condiciones normales de repetición.

**Atención:** Las siguientes funciones POKE no trabajan en modo directo, deben ser utilizadas dentro de un programa.

POKE 199,1; La impresión se realiza en modo invertido.

POKE 199,0; La impresión retorna al modo normal.

POKE 204,0; El cursor parpadea durante un comando GET.

POKE 204,1-255; Elimina el anterior.

POKE 211,0-21; Sitúa el cursor en una columna determinada.

POKE 214,0-22; Sitúa el cursor en una línea.

Para terminar, una advertencia: el uso indiscriminado de funciones POKE — aunque se cometan equivocaciones — no puede causar ningún tipo de averías en su equipo; solamente si tiene un programa cargado en el VIC y comete un error, es posible que se pierda, pues en determinados casos el error cometido sólo será posible subsanarlo desconectando el ordenador de la corriente, con lo que dicho programa se perderá si no se ha copiado antes. ■





## NUMEROLOGÍA NO ESOTÉRICA

# un calendario perpetuo



**PARA LOS QUE NO SABEN EN QUÉ DÍA VIVEN Y PARA LOS QUE, SABIÉNDOLO, SIENTEN CURIOSIDAD POR CONOCER SU EDAD EN DÍAS: ¡AHÍ VA ESTE PROGRAMA!**

**P. MASATS**

En la figura 1 se da el listado del programa y en la 2 una muestra de los resultados. Aviso a los curiosos que han dejado atrás la adolescencia: si calcula su edad en días, el resultado estará siempre por encima de sus expectativas. He aquí una muestra palpable de lo que algunos llaman «la estupidez de las máquinas».

```
10 REM CALENDARIO PERPETUO
20 REM P. MASATS; MICROELECTRONICA Y CON
TROL
30 PRINT"♥"
40 GOSUB960
50 PRINT:PRINT
60 PRINT"**********************"
70 PRINT"*CALCULADOR DE FECHAS*"
80 PRINT"**********************"
90 PRINT
100 PRINT"AVERIGUAR:"
110 PRINT
120 PRINT"1) EL NUMERO DE DIAS"
130 PRINT
140 PRINT"   ENTRE DOS FECHAS"
150 PRINT
160 PRINT"2) EL DIA DE LA SEMANA"
170 PRINT
180 PRINT"ENTRE SU OPCION"
190 PRINT
200 INPUT A
210 PRINT
220 IF A<1 OR A>2 THEN PRINT"LAS OPCIONE
S SON 1 Y 2 !":GOTO190
230 PRINT
240 ON A GOSUB 320,530
250 PRINT
260 INPUT"HA TERMINADO";A$
270 A$=LEFT$(A$,1)
280 IF A$="N" THEN RUN
290 IF A$<>"S" THEN 250
300 PRINT"♥"
310 END
320 PRINT"♥"
330 GOSUB740
340 FORJ=1TO12
350 M(J)=M(J)+M(J-1)
360 NEXTJ
370 GOSUB1090
380 D1=DIA
390 GOSUB740
400 GOSUB1090
410 PRINT"♥"
420 PRINT"EL NUMERO DE DIAS "
430 PRINT
440 PRINT"ENTRE LAS DOS"
450 PRINT
460 PRINT"FECHAS ES";ABS(D1-DIA)
470 PRINT
480 PRINT
490 PRINT"PULSE UNA TECLA"
500 PRINT
510 GETA$:IF A$=""THEN510
520 RETURN
530 PRINT"♥"
540 GOSUB740
550 GOSUB670
560 PRINT"♥"
570 PRINT
580 PRINT"EL ";DIA;" DE ";M$(ME)
590 PRINT
600 PRINT"DE ";AN+SIG*100
610 PRINT
620 PRINT"ES UN ";DIA$(I)
630 PRINT
640 PRINT"PULSE UNA TECLA"
650 GETA$:IFA$=""THEN650
660 RETURN
670 REM CALCULO DEL DIA DE LA SEMANA
680 ME=ME-2
690 IFME<1THENME=ME+12:AN=AN-1
700 I=INT(2.6*ME-.19)+DIA+AN+INT(AN/4)+I
NT(SIG/4)-2*SIG
710 I=I-INT(I/7)*7
720 ME=ME+2
730 RETURN
740 REM ENTRADA DE FECHA
750 PRINT
760 PRINT"ENTRE LA FECHA EN NUM."
770 PRINT
780 PRINT"POR EJEMPLO:"
790 PRINT
800 PRINT"31,12,1982"
810 PRINT
820 INPUT DIA,ME,AN
830 BISI=0
840 SIG=INT(AN/100)
850 AN=AN-SIG*100
860 IF(AN-INT(AN/4)*4)=0 OR (SIG-INT(SIG
/4)*4=0) THEN BISI=1
870 IFM(ME)=>DIA AND DIA>0 THEN GOTO 950
880 IF ME=2 AND BISI=1 AND DIA=29 THEN G
OTO 950
890 PRINT
900 PRINT"NO EXISTE ESTE DIA EN"
910 PRINT
920 PRINT"EL MES DE ";M$(ME)
930 PRINT
940 GOTO760
950 RETURN
960 REM INICIALIZACION
970 DIMM$(12),M(12),DIA$(6)
980 DATA ENERO,31,FEBRERO,28,MARZO,31,AB
RIL,30,MAYO,31
990 DATA JUNIO,30,JULIO,31,AGOSTO,31,SET
IEMBRE,30
1000 DATA OCTUBRE,31,NOVIEMBRE,30,DICIEM
BRE,31
1010 DATA DOMINGO,LUNES,MARTES,MIERCOLES
,JUEVES,VIERNES,SABADO
1020 FORI=1TO12
1030 READM$(I),M(I)
1040 NEXTI
1050 FORI=0TO6
1060 READ DIA$(I)
1070 NEXTI
1080 RETURN
1090 REM CALCULO DIAS
1100 AN=AN+SIG*100
1110 DIA=DIA+M(ME)+365*AN+INT((AN-1)/4)
1120 IFINT(AN/4)*4=AN AND ME>2 THEN DIA=
DIA-1
1130 RETURN

READY.
```

Fig. 1

```
**********************
*CALCULADOR DE FECHAS*
**********************
AVERIGUAR:

1) EL NUMERO DE DIAS

   ENTRE DOS FECHAS

2) EL DIA DE LA SEMANA

ENTRE SU OPCION

? 1

ENTRE LA FECHA EN NUM.

POR EJEMPLO:

31,12,1982

? 1 , 1 , 1900

ENTRE LA FECHA EN NUM.

POR EJEMPLO:

31,12,1982

? 2 , 8 , 1982

EL NUMERO DE DIAS

ENTRE LAS DOS

FECHAS ES 30164

PULSE UNA TECLA

HA TERMINADO?NO
```

```
**********************
*CALCULADOR DE FECHAS*
**********************
AVERIGUAR:

1) EL NUMERO DE DIAS

   ENTRE DOS FECHAS

2) EL DIA DE LA SEMANA

ENTRE SU OPCION

? 2


ENTRE LA FECHA EN NUM.

POR EJEMPLO:

31,12,1982

? 2 , 8 , 1982

EL  2  DE AGOSTO

DE  1982

ES UN LUNES

PULSE UNA TECLA

HA TERMINADO?SI
```

Fig. 2

## VENTANA CBM

# protección de programas

## inhibición de la tecla "STOP"

por **JOAN CARLES SAMARANCH**

El disponer de una tecla STOP es realmente muy interesante durante el «debuging» de los programas; pero, cuando los programas están funcionando definitivamente, el pulsarla por equivocación, sobre todo por una persona distinta a la que ha realizado el programa, puede ser un engorro.

Existen dos formas distintas de inhibir dicha tecla: una fácil y otra más complicada.

Aunque estos ejemplos sean sencillos implican varios conceptos de cierto nivel, que podemos aprovechar desde un punto de vista didáctico, tales como: la instrucción POKE, el concepto de página cero e incluso una pequeña parte de código máquina.

A grandes rasgos, la función de la instrucción POKE consiste en dar a una posición de memoria un determinado valor. En principio parece sencillo y lo es. Lo realmente complicado es saber la función de cada posición de memoria y qué valores darle.

En nuestro caso vamos a trabajar con dos posiciones de memoria muy concretas: 144 y 145 (BASIC 4.0), en las cuales está definido el vector de interrupción (IRQ). Dicho vector apunta a una determinada dirección dentro de la ROM (S.O.) que lo primero que hace es llamar a una subrutina en código máquina (esta instrucción ocupa 3 bytes), que se encarga de actualizar la variable TI$ (reloj interno) y comprobar si se ha pulsado la tecla STOP.

Si sumamos 3 a la dirección comentada anteriormente, lo que haremos será prescindir de la subrutina de consulta de la tecla STOP, inhibiéndola. Un efecto secundario del cambio es la pérdida de actualización del reloj (TI$) que si no se usa no tiene mayor importancia.

Veamos a continuación una tabla de lo dicho para los distintos equipos:

| | BASIC 1.0 | BASIC 2.0 | BASIC 3.0 |
|---|---|---|---|
| PEEK (50003)= | 0 | 1 | 160 |
| Inhibir STOP | | | |
| POKE | 537,136 | 144,49 | 144,88 |
| Desinhibir STOP | | | |
| POKE | 537,133 | 144,46 | 144,85 |

En caso de necesitar la variable TI$ implementaremos un pequeño programa en código máquina que, interfiriendo el IRQ, active la subrutina de reloj pero que anule la detección de la tecla de STOP poniendo $FF en la posición $9B (155 en decimal) de página cero.

Para BASIC 4.0 rodar el siguiente programa:

```
100 D=852:D$="20>:??:9??8=9;
  004<58>4"
110 FOR J=1 TO LEN(D$)/2
120 POKEJ+D, ASC(MID$(D$,J*2-1)
  )*16
+ASC(MID$(D$,J*2))-816
130 NEXT
```

Después activarlo con POKE 145,3 y desactivarlo con POKE 145,228.

Para BASIC 2.0 cambiar la sentencia:

```
100 D=844:D$="20>:??:9??8=9;
  004<31>6"
```

activar con POKE 144,77 : POKE 145,3 y desactivar con POKE 145,230 : POKE 144,46.

Para BASIC 1.0 cambiar la sentencia:

```
100 D=900:D$="20>:??:9??8=
  09024<88>6"
```

activar con POKE 538,3 y desactivar con POKE 538,230.

**POKES para el VIC-20**

También es posible para el VIC-20 proteger la tecla de STOP, dejando de funcionar TI/TI$, con el POKE 788,194. Para reactivar esta tecla escribir POKE 788,171.

Otra tecla interesante de proteger es la de RESTORE con POKE 37150,2 y reactivar con POKE 37150,130.

Y por último el efecto de cambio mayúsculas/minúsculas con las teclas CBM+shift se puede evitar con PRINT CHR$(8) y restaurar con PRINT CHR$ (9). ■

## un programa de utilidad

```
10 LK=197:SH=653
20 IFPEEK(LK)=39THENSTOP
30 IFPEEK(SH)THEN70
40 POKE 36864,255 AND PEEK(36864)-(PEEK(
LK)=23)
50 POKE 36865,255 AND PEEK(36865)-(PEEK(
LK)=31)
60 GOTO20
70 POKE 36864,255 AND PEEK(36864)+(PEEK(
LK)=23)
80 POKE 36865,255 AND PEEK(36865)+(PEEK(
LK)=31)
90 GOTO20

READY.
```

Fig. 1

**En los juegos en cartucho existe la posibilidad de centrar el cuadrado de información dentro de la pantalla del televisor. Para poder disfrutar de esta facilidad en nuestros programas puede usarse el programa listado en la figura 1. Pulsando las teclas de movimiento del cursor, conjuntamente con las de SHIFT, en caso necesario obtendremos los cuatro movimientos básicos. Cuando la pantalla esté en el sitio deseado, pulsaremos F1 para salir del programa. Si sustituimos el STOP de la línea 20 por un número de línea podremos iniciar la ejecución del programa principal (que podrá empezar a partir de la línea 100).**

## MAPA DE MEMORIA DEL VIC-20 (II)

# una ojeada al VIC propiamente dicho

## (VIDEO INTERFACE CHIP)

La característica más aparente del VIC-20 (la que, realmente, salta más a la vista) es la posibilidad de hacer gráficos con una cierta sofisticación, siendo el responsable de ello (y de más de un dolor de cabeza del sufrido usuario) un circuito integrado llamado «chip de interface de video». La manera cómo la CPU 6502 —y nosotros, a través de ella— controla a este circuito va a ser el tema de este artículo (que para más comodidad del lector va a publicarse en varias partes).

### EL «MEMORY MAPPED» O MAPEADO DE MEMORIA

El sistema que utiliza la CPU para «decirle» al interface de video (en adelante para abreviar vamos a llamarle «vic» en minúsculas para distinguirlo de nuestro ordenador), qué información y cómo ha de «sacar» esta información por la pantalla, es el denominado «memory mapped» o mapeado por memoria (ya sé que suena raro, pero es la traducción más o menos literal) que quiere decir que los registros del vic donde la CPU «escribe» la información necesaria para manejar los datos (a esta información también se le da el nombre de «parámetros de video») o «lee» el estado del vic en determinado momento, le aparece como una serie de posiciones de memoria absolutamente iguales a las de RAM y que tienen una especie de puerta trasera por la que el vic tiene acceso a esta información.

### LOS 16 REGISTROS DEL VIC Y SUS FUNCIONES

El número de los registros del vic es de dieciséis, son consecutivos y ocupan las direcciones de memoria 36864-36879 en decimal y $9000-$900F en Hex. Obviamente al ser estos registros posiciones de memoria podemos acceder a la información que contienen mediante la instrucción PEEK y modificar dicha información con POKE. Los diferentes registros y sus funciones se analizan a continuación:

**Reg. 0. - 36864 dec. - $9000**

Un valor entre 0 y 127 determina la posición del borde izquierdo de la pantalla. El valor normal es 12. Pruebe el programa siguiente:

```
10 FOR J=5 TO 30:POKE 36864,J:
NEXT J:FOR J=30 TO 5 STEP 1:POKE
36864,J:NEXT J
```

Si se añade 128 al valor del Reg. 0, la información de video pasa al modo «entrelazado». En la mayoría de los casos no hay cambios si se hace POKE 36864,140. Este modo de video tiene relación con el funcionamiento interno de algunos televisores.

**Reg. 1. - 36865 dec. - $9001**

El valor de este registro determina la posición del borde superior. Puede variar entre 0 y 255. El valor normal es 38. Para deslizar la pantalla en sentido vertical:

```
20 FOR J=25 TO 45:POKE 36865,J:
NEXT J:FOR J=45 TO 25 STEP —1:
POKE 36865,J:NEXT J
```

**Reg. 2. - 36866 dec. - $9002**

Parte de este registro se dedica a definir el número de columnas con que opera la pantalla. Normalmente son 22, valor al que se debe sumar 128 para obtener el contenido de este registro. 128 se usa aquí para trabajar normalmente. Con un valor de 22 trabajaremos en el modo llamado «pantalla alternativa» del que hablaremos más adelante. De momento diremos que es útil para hacer un cambio rápido de una pantalla a otra, para animación, etc...

**Reg. 3. - 36867 dec. - $9003**

Un registro con mucho trabajo. Siempre está cambiando. Pruebe a hacer ?PEEK(36867) unas cuantas veces y le dará diferentes valores 46, 174... De momento réstele 128, si es mayor que este número, con lo que tenemos siempre 46, que es la cantidad que normalmente encontraremos en este registro y que es, a su vez, el número de líneas, multiplicado por dos, que tiene la pantalla (o sea: 23 * 2 = 46).

Hay otra cosa que se maneja en este registro y es importante: si se suma 1 al valor existente, el generador de caracteres cambia a otra configuración; la de doble carácter, es decir que el carácter que teclee ocupará el doble de espacio en la pantalla.

Esto no es completamente automático. Si le decimos al vic que dibuje caracteres dobles debemos también decirle cómo deben dibujarse estos signos, así que prepárese para ver cosas raras en el siguiente experimento. Esto pasará porque no hemos preparado nuevos caracteres para trabajar con este nuevo modo.

Teclee POKE 36867,47: la pantalla se convertirá en una serie de signos confusos. No se preocupe de momento por esto. Teclee ahora SHIFT CLR/

## UNO DE LOS CIRCUITOS INTEGRADOS MÁS POTENTES Y VERSÁTILES DEL VIC-20 ES EL QUE DA NOMBRE AL EQUIPO. AQUÍ EMPEZAMOS A TRATAR DE SUS REGISTROS Y DE CÓMO MANEJARLO

por **PERE MASATS**

HOME. La pantalla se «limpia» pero el cursor aparece algo extraño. Tampoco debe preocuparse. En seguida vamos a ver qué pasa.

El primer carácter en la tabla del VIC es el que corresponde al signo «@», el siguiente es «A» y luego «B», y así sucesivamente. Ahora pulse la tecla «@»: en vez de aparecer el primer carácter solo, nos aparecen los dos primeros, el «@» y el «A», uno sobre el otro. Pulse ahora la letra «A» y verá que aparecen las letras «B» y «C» que son los siguientes en la lista de caracteres.

¿Qué está pasando? Cada carácter ahora ocupa el doble de espacio en la pantalla y, por lo tanto, el vic «busca» el doble de información en el generador de caracteres (que es el nombre con el que se conoce la tabla que contiene las configuraciones de cada carácter individual). Como la tabla no ha cambiado «coge» dos caracteres consecutivos.

Cuando decida usar esta característica debe, primero, diseñar su propio generador de caracteres de acuerdo con las nuevas configuraciones. La posibilidad de escribir caracteres dobles se usa con frecuencia en gráficos de alta resolución. Los elementos de la tabla controlarán, individualmente, cada punto (dot o pixel, en inglés) del dibujo.

Puede usted volver a los buenos viejos tiempos tecleando POKE 36867,46, pero si le resulta difícil, dado el estado de la pantalla, simplemente cierre el interruptor de red y vuelva a abrirlo.

### Reg. 4. - 36868 dec. - $9004

Este registro cambia continuamente. Junto con el bit más significativo del registro anterior (el valor 128 que despreciábamos antes) forma una cantidad que nos indica qué línea se está «barriendo» en la pantalla en el momento de la lectura. Dicha cantidad varía tan rápidamente que no es práctico utilizarla en Basic.

### Reg. 5. - 36869 dec. - $9005

Éste es un registro muy importante. Controla la localización de dos tablas: la que contiene los caracteres de la pantalla y la que contiene la configuración de cada uno de los caracteres que se pueden representar (el «generador de caracteres»). Vamos a dedicarnos a cada una por separado.

La tabla de pantalla contiene 506 caracteres (22 * 23) que se exhiben en la pantalla ocupando cada uno una posición de memoria. Vamos a calcular dónde empieza dicha tabla:

Lea el contenido de 36869 ( ?PEEK (36869) ), divídalo por 16 y olvídese del resto de la división. Debe darle un número entre 8 y 15. Réstele 8 y multiplique el resultado por 2, debiendo resultar una cantidad entre 0 y 14. Ahora bien, si el contenido de 36866 es 128, o mayor, súmele 1 al valor anterior y multiplique el resultado por 512. En este punto debe tener un número entre 0 y 7680. Ésta es la posición del primer byte de la memoria de pantalla que normalmente será 7680 pero que si Ud. trabaja con más de 8 K de RAM será 4096 (¡¡anote en sitio MUY visible estas cantidades pues es una de las fuentes de problemas más frecuentes que hemos observado!!).

Hemos llegado a una cierta conclusión: a través del Reg. 5 y parte del 2,

(pasa a la pág. siguiente)

# mapa de memoria del VIC-20

(viene de la pág. anterior)

podemos elegir la posición de nuestra pantalla con ciertas limitaciones: debe estar entre 0 y 7680 y ser múltiplo de 512. Si quiere resituar su pantalla haga el cálculo al revés. Divida la dirección escogida por 512, réstele 1 si es impar, divídalo por 2, súmele 8 y finalmente multiplique por 16. (¡UFF!) Podemos ver aquí que el bit más significativo de la dirección de la pantalla es el valor 128 «añadido» al Reg. 2, por lo que justifica el nombre de «pantalla alternativa».

El generador de caracteres también se localiza mediante este registro. Si necesitamos definir nuestro propio juego de caracteres (no sólo para dibujar gráficos sino también para poder escribir con caracteres que no forman parte del alfabeto inglés: la ñ del castellano, la ç del catalán, etc...) debemos cambiar la parte que le afecta. El cálculo de la dirección es como sigue:

Lea el contenido del registro, divídalo por 16 y ahora tome el resto — no el cociente — y si es mayor que 7 réstele 8. Por otra parte, si el resto no es mayor que 7 súmele 32. A estas alturas tendrá un número que será o bien 7 o entre 32 y 39.

Multiplique ahora este número por 1024 y tendrá la dirección del generador de caracteres que está manejando el vic. Deberá estar entre 0 y 7168 o entre 32768 (la posición normal) y 39936. Y debe ser múltiplo de 1024. Si quiere Ud. definir su propio juego de caracteres deberá situarlo en una zona de RAM, entre 0 y 7168. En este caso, haga el cálculo al revés: defina la dirección, divídala por 1024, súmele 8 y ya tiene el resultado.

No se olvide que las direcciones de la memoria de pantalla y del juego de caracteres se «empaquetan» juntas en este registro y que sus nombres oficiales son «matriz de video» y «generador de caracteres».

### PROGRAMA QUE FACILITA LOS PARÁMETROS DE VIDEO MÁS IMPORTANTES

Para facilitarle el trabajo con éste y otros registros se incluye un programa (fig. 1) que, en forma de subrutina, le entrega los parámetros de video más importantes manejados por el vic. La inclusión de esta subrutina en un programa que maneje la matriz de video mediante POKEs le permitirá hacer que dicho programa se adapte automáticamente a cualquier expansión de memoria RAM haciendo que los POKEs se realicen sobre la variable AV más o menos como sigue:

```
1 GOSUB 7000
..........................
100 IF AV = 7680 THEN CV = 38400
110 IF AV = 4096 THEN CV = 37888
120 FOR I = 0 TO 506
130 POKE AV + I, 160
140 POKE CV + I, 0
150 NEXT I
```

```
10 GOSUB7000
20 END
7000 AV=PEEK(36869):AV=INT(AV/16):AV=(AV
-8)*2
7010 IFPEEK(36866)>=128THENAV=AV+1
7020 AV=AV*512
7030 PRINT"A.V.=";AV
7040 NL=PEEK(36867):IFNL>=128THENNL=NL-1
28
7050 IFNL/2=INT(NL/2)THENCS=8
7060 IFNL/2<>INT(NL/2)THENCS=16
7070 NL=INT(NL/2)
7075 PRINT"N. L.=";NL
7080 PRINT"CAR.=":CS;"X8"
7090 AC=PEEK(36869):AC=AC/16:AC=AC-INT(A
C)
7100 IFAC>7THENAU=0
7110 IFAC<=7THENAU=1
7120 IFAU=0THENAC=AC-8
7130 IFAU=1THENAC=AC+32
7140 AC=AC*1024
7145 PRINT"CH. GEN.=";AC
7150 NC=PEEK(36866):IFNC>=128THENNC=NC-1
28
7160 PRINT"N. COL.=":NC
7170 RETURN

READY.
```

Fig. 1

En el ejemplo anterior vemos un fragmento de programa (se supone que la rutina de la figura 1 está en su sitio) que calcula la posición de la memoria de color en función de la situación de la matriz de video (las dos cambian con la expansión de la memoria pero en sentidos opuestos) en las líneas 100 y 110, entre 120 y 150 «pinta» la pantalla de negro pero lo hace acoplando el programa automáticamente a la configuración de memoria existente en el momento de la ejecución.

**Reg. 6 y 7. - 36870 y 36871 dec. - $9006 y $9007**

Aquí tiene su lápiz luminoso. No se trata de un lápiz con un láser dentro sino de un dispositivo que le permite «señalarle» un punto de la pantalla al VIC y luego por programa hacer, por ejemplo, dibujos «a mano». Estos registros permiten saber a qué punto estamos «apuntando».

Se pueden leer las coordenadas X e Y del punto en los registros 6 y 7 respectivamente. Las cantidades que entregan estos registros varían entre 0 y 255. Vigile para que no se produzcan oscilaciones en la posición del lápiz. Si éstas son inevitables, el programa deberá realizar algún tipo de promediado de las lecturas para evitarlas. Otro método es el llamado de histéresis: un nuevo valor se ignora si difiere del valor previo en menos de una determinada cantidad.

**Reg. 8 y 9. - 36872 y 36873 dec. - $9008 y $9009**

Aquí se pueden leer las posiciones de los potenciómetros de juegos (también llamados raquetas o en inglés «paddles»). Alguna parte del recorrido del potenciómetro puede quedar sin lectura (es decir: puede que leamos el valor 0 ó 255 antes de llegar al extremo respectivo). Asimismo debe vigilarse la aparición de oscilaciones del mismo modo que con el lápiz luminoso. Como información complementaria debemos decir que se puede conectar al VIC-20 una palanca de juegos («Joystick» en inglés). Sin embargo, el vic no se encarga de su manejo, que corre a cargo de las direcciones 37151 y 37152.

(continuará)



# CORREO ABIERTO

*Tal como decíamos en la presentación de esta Sección, aún no contamos con consultas escritas llegadas a esta Revista. Sabemos que no faltan las ganas de preguntar porque no han sido pocos los usuarios que nos han planteado sus dudas aprovechando las ocasiones y los medios más diversos. Para que otros se animen y por estimar que las respuestas que ya hemos dado pueden interesar a otros, además de aquellos que las han recibido de viva voz, ahí van unas cuantas de las que nos han parecido seleccionables. Esperamos que sirvan para aportar luz en medio de la oscuridad de las dudas...*

**Pregunta: ¿Puede utilizarse un VIC para controlar un tren de juguete?**

**Respuesta:** ¡Por supuesto! Para dar sólo un ejemplo muy conciso, a través del port de usuario y con muy pocos componentes, puede usted controlar hasta diez agujas directamente. Sofisticando un poco más el programa y el interface (el montaje que relaciona el ordenador con el tren propiamente dicho) se puede llegar a aquello tan típico del trabajo con ordenadores: el único límite es la imaginación (y perdón por el tópico).

•

**Pregunta: Más o menos tengo una idea clara de lo que significan las palabras hardware y software pero... ¿qué es el firmware?**

**Respuesta:** Empecemos por aclarar lo del hard. y el soft. y hagámoslo con un símil: imaginemos que hemos adquirido en una tienda una cassette sin grabar. En este caso tenemos el hardware, es decir el medio material sobre el que registraremos una pieza musical, una conversación, etc... En

(pasa a la pág. siguiente)

## MARKETCLUB

*La posibilidad de realizar ventas, compras, intercambios, constituye un medio de intercomunicación entre los que comparten una misma afición, una misma actividad, idénticos afanes. Para fomentar en el seno del "Club Commodore" las relaciones entre sus miembros, queda abierta esta Sección en la que se dará buena acogida a los textos de ofertas o peticiones relativas a los diferentes modelos de micro-ordenadores "Commodore", a sus periféricos, a programas, a libros, a información... Extensión máxima por comunicación: cincuenta palabras. Aquí se incluyen las primeras que pueden servir de ejemplo para animar a los que tengan algo que ofrecer o a los que quieran pedir algo.*

- **VENDO Compilador de BASIC PET-SPEED 2.0 para CBM 8032-8050, optimiza la velocidad un 50 %, prácticamente nuevo. Razón: José Luis Villalobos. Teléfono (93) 200 43 69.**
- **INTERCAMBIARÍA programas para el PET 2001 o CBM 3000. Ofertas: Santi EA3BVT. Tel. 246 04 65. Apart. 5.350. Barcelona.**

## CORREO ABIERTO (viene de la pág. anterior)

los ordenadores, cuando hablamos del hardware nos referimos al conjunto de componentes electrónicos que constituyen la parte material (tranquilos, aún no se ha demostrado que también tengan espíritu, aunque a veces parezca que tienen muy mala idea) del equipo; así decimos, que determinada función la hacemos «por hardware» cuando montamos un conjunto de circuitos para realizar esta función. Si en la cassette anterior registramos un programa de radio, entonces podemos decir que le hemos incorporado el software, es decir, le hemos incorporado una parte no material (propiamente: una información) que hace a esta cassette diferente de otra con una canción grabada. En lo que respecta a los ordenadores, el soft. es un programa o conjunto de programas que hacen que el hard. se convierta en un ordenador. De hecho, muy a menudo el mismo hardware, con distintos softwares, realiza trabajos completamente diferentes, lo que constituye la razón de la versatilidad de los ordenadores.

El firmware es algo intermedio entre los dos conceptos anteriores; imaginemos que, como en Babilonia, escribimos haciendo incisiones en una tablilla de arcilla. Como en el ejemplo anterior, la tablilla en sí misma es el hardware y el texto es el software, pero ahora las incisiones hacen que la tablilla haya cambiado de forma físicamente, con lo que ambos conceptos se confunden y podemos hablar de software por hardware. Pues bien, esto es el firmware. En los ordenadores VIC y CBM, el Kernal y el interpretador de Basic (que son programas o conjuntos de programas) van incorporados en unos circuitos integrados llamados ROM (de Read Only Memory o memoria de lectura solamente, es decir que sólo se puede leer) y a los que durante el proceso de fabricación se les incorpora el programa.

●

**Pregunta: Tengo algún juego en cartucho para el VIC-20. Si desconecto el ordenador y lo vuelvo a conectar, el programa empieza a funcionar automáticamente sin tener que hacer RUN como en los programas normales. ¿Cómo se consigue esto? Y ¿puedo conseguir lo mismo con mis programas en cinta?**

**Respuesta:** Todos los micro-ordenadores tienen una rutina llamada — generalmente — de inicialización, la cual se ejecuta tan pronto como se conecta la corriente. Esta rutina verifica cuanta memoria hay conectada (y exhibe el mensaje XXXX BYTES FREE como resultado), pone los valores adecuados en varias posiciones de memoria (para uso de otros programas) y pasa al Interpretador de Basic para esperar una instrucción desde el teclado. En el VIC-20 durante la ejecución de esta rutina se verifica que las posiciones de memoria $A004, $A005, $A006, $A007 y $A008 contengan determinados datos (en este caso los valores hexadecimales: 41, 30, C3, C2 y CD). Si esto es así, se ejecuta el programa contenido en la ROM del juego, dado que se ha comprobado que está conectada. Si no se leen estos valores en las posiciones de memoria indicadas, ello se interpreta como una comprobación de que no hay un programa de ejecución automática conectado. Entonces se continúa con la rutina de inicialización y, en su momento, se salta al Basic y así tenemos el VIC funcionando normalmente. Como se ve, este sistema no sirve para el cassette. No obstante, circulan rumores en torno a la posibilidad de que, bajo ciertas circunstancias, y si se reúnen determinadas condiciones... (total: a lo mejor sí).

●

**Pregunta: En un programa tengo que entrar un valor numérico con una instrucción INPUT y un mensaje y, al ejecutarlo, me contesta: ¿REDO FROM START? ¿Puede decirme qué pasa?**

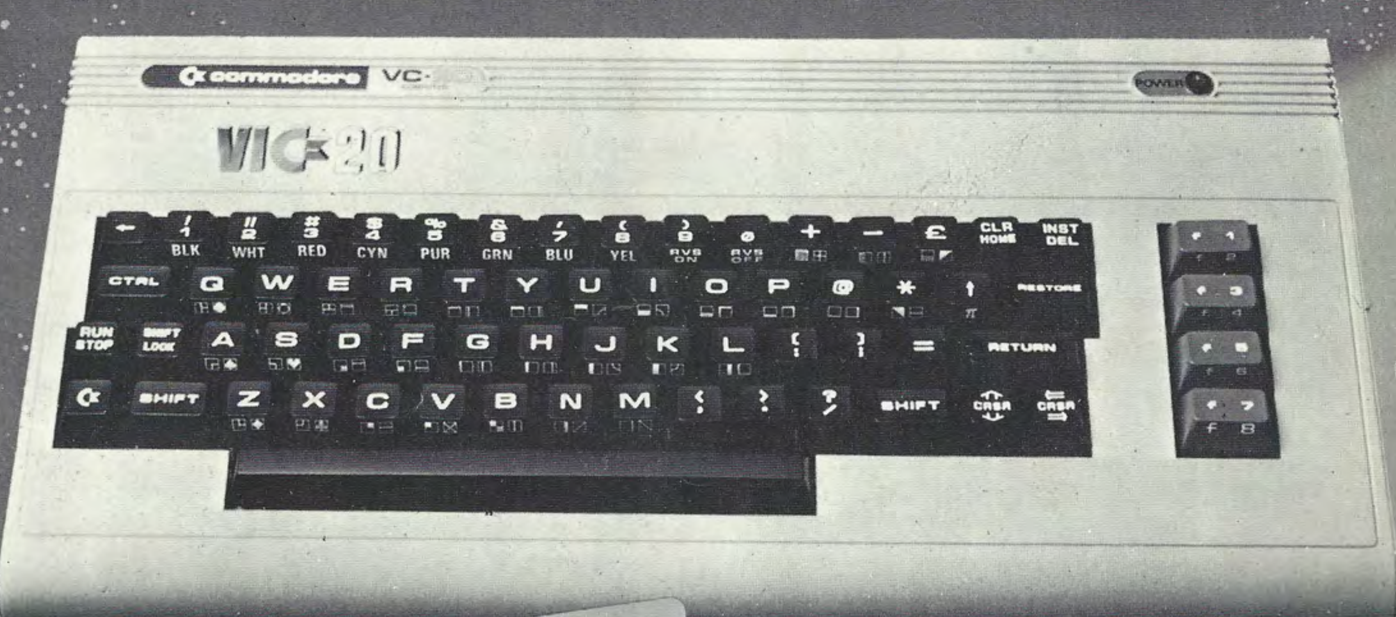
**Respuesta:** Por lo que me dice, puedo sugerirle que reduzca la longitud del mensaje a menos de 20 caracteres o, en caso de no poder hacerlo, introduzca, antes del comando INPUT, un PRINT que le dé el mensaje, pues es una característica del Basic del VIC (del Basic V2 en concreto) el no aceptar los datos si el mensaje es mayor de esta longitud.

| Hex | Decimal | Description |
|---|---|---|
| 0000-0002 | 0-2 | USR jump |
| 0003-0004 | 3-4 | Float-Fixed vector |
| 0005-0006 | 5-6 | Fixed-Float vector |
| 0007 | 7 | Search character |
| 0008 | 8 | Scan-quotes flag |
| 0009 | 9 | TAB column save |
| 000A | 10 | 0=LOAD, 1=VERIFY |
| 000B | 11 | Input buffer pointer/# subscrpt |
| 000C | 12 | Default DIM flag |
| 000D | 13 | Type: FF=string, 00=numeric |
| 000E | 14 | Type: 80=integer, 00=floating point |
| 000F | 15 | DATA scan/LIST quote/memry flag |
| 0010 | 16 | Subscript/FNx flag |
| 0011 | 17 | 0=INPUT;$40=GET;$98=READ |
| 0012 | 18 | ATN sign/Comparison eval flag |
| 0013 | 19 | Current I/O prompt flag |
| 0014-0015 | 20-21 | Integer value |
| 0016 | 22 | Pointer: temporary strg stack |
| 0017-0018 | 23-24 | Last temp string vector |
| 0019-0021 | 25-33 | Stack for temporary strings |
| 0022-0025 | 34-37 | Utility pointer area |
| 0026-002A | 38-42 | Product area for multiplication |
| 002B-002C | 43-44 | Pointer: Start-of-Basic |
| 002D-002E | 45-46 | Pointer: Start-of-Variables |
| 002F-0030 | 47-48 | Pointer: Start-of-Arrays |
| 0031-0032 | 49-50 | Pointer: End-of-Arrays |
| 0033-0034 | 51-52 | Pointer: String-storage(moving down) |
| 0035-0036 | 53-54 | Utility string pointer |
| 0037-0038 | 55-56 | Pointer: Limit-of-memory |
| 0039-003A | 57-58 | Current Basic line number |
| 003B-003C | 59-60 | Previous Basic line number |
| 003D-003E | 61-62 | Pointer: Basic statement for CONT |
| 003F-0040 | 63-64 | Current DATA line number |
| 0041-0042 | 65-66 | Current DATA address |
| 0043-0044 | 67-68 | Input vector |
| 0045-0046 | 69-70 | Current variable name |
| 0047-0048 | 71-72 | Current variable address |
| 0049-004A | 73-74 | Variable pointer for FOR/NEXT |
| 004B-004C | 75-76 | Y-save; op-save; Basic pointer save |
| 004D | 77 | Comparison symbol accumulator |
| 004E-0053 | 78-83 | Misc work area, pointers, etc |
| 0054-0056 | 84-86 | Jump vector for functions |
| 0057-0060 | 87-96 | Misc numeric work area |
| 0061 | 97 | Accum#1: Exponent |
| 0062-0065 | 98-101 | Accum#1: Mantissa |
| 0066 | 102 | Accum#1: Sign |
| 0067 | 103 | Series evaluation constant pointer |
| 0068 | 104 | Accum#1 hi-order (overflow) |
| 0069-006E | 105-110 | Accum#2: Exponent, etc. |
| 006F | 111 | Sign comparison, Acc#1 vs #2 |
| 0070 | 112 | Accum#1 lo-order (rounding) |
| 0071-0072 | 113-114 | Cassette buff len/Series pointer |
| 0073-008A | 115-138 | CHRGET subroutine; get Basic char |
| 007A-007B | 122-123 | Basic pointer (within subrtn) |
| 008B-008F | 139-143 | RND seed value |
| 0090 | 144 | Status word ST |
| 0091 | 145 | Keyswitch PIA: STOP and RVS flags |
| 0092 | 146 | Timing constant for tape |
| 0093 | 147 | Load=0, Verify=1 |
| 0094 | 148 | Serial output: deferred char flag |
| 0095 | 149 | Serial deferred character |
| 0096 | 150 | Tape EOT received |
| 0097 | 151 | Register save |
| 0098 | 152 | How many open files |
| 0099 | 153 | Input device, normally 0 |
| 009A | 154 | Output CMD device, normally 3 |
| 009B | 155 | Tape character parity |
| 009C | 156 | Byte-received flag |
| 009D | 157 | Direct=$80/RUN=0 output control |
| 009E | 158 | Tp Pass 1 error log/char buffer |
| 009F | 159 | Tp Pass 2 err log corrected |
| 00A0-00A2 | 160-162 | Jiffy Clock HML |
| 00A3 | 163 | Serial bit count/EOI flag |
| 00A4 | 164 | Cycle count |
| 00A5 | 165 | Countdown,tape write/bit count |
| 00A6 | 166 | Tape buffer pointer |
| 00A7 | 167 | Tp Wrt ldr count/Rd pass/inbit |
| 00A8 | 168 | Tp Wrt new byte/Rd error/inbit cnt |
| 00A9 | 169 | Wrt start bit/Rd bit err/stbit |
| 00AA | 170 | Tp Scan;Cnt;Ld;End/byte assy |
| 00AB | 171 | Wr lead length/Rd checksum/parity |
| 00AC-00AD | 172-173 | Pointer: tape bufr, scrolling |
| 00AE-00AF | 174-175 | Tape end adds/End of program |
| 00B0-00B1 | 176-177 | Tape timing constants |
| 00B2-00B3 | 178-179 | Pntr: start of tape buffer |
| 00B4 | 180 | 1=Tp timer enabled; bit cnt |
| 00B5 | 181 | Tp EOT/RS232 next bit to send |
| 00B6 | 182 | Read character error/outbyte buf |
| 00B7 | 183 | # characters in file name |
| 00B8 | 184 | Current logical file |
| 00B9 | 185 | Current secndy address |
| 00BA | 186 | Current device |
| 00BB-00BC | 187-188 | Pointer to file name |
| 00BD | 189 | Wr shift word/Rd input char |

| Hex | Decimal | Description |
|---|---|---|
| 00BE | 190 | # blocks remaining to Wr/Rd |
| 00BF | 191 | Serial word buffer |
| 00C0 | 192 | Tape motor interlock |
| 00C1-00C2 | 193-194 | I/O start adds |
| 00C3-00C4 | 195-196 | Kernel setup pointer |
| 00C5 | 197 | Last key pressed |
| 00C6 | 198 | # chars in keybd buffer |
| 00C7 | 199 | Screen reverse flag |
| 00C8 | 200 | End-of-line for input pointer |
| 00C9-00CA | 201-202 | Input cursor log (row, column) |
| 00CB | 203 | Which key: 64 if no key |
| 00CC | 204 | 0=flash cursor |
| 00CD | 205 | Cursor timing countdown |
| 00CE | 206 | Character under cursor |
| 00CF | 207 | Cursor in blink phase |
| 00D0 | 208 | Input from screen/from keyboard |
| 00D1-00D2 | 209-210 | Pointer to screen line |
| 00D3 | 211 | Position of cursor on above line |
| 00D4 | 212 | 0=direct cursor, else programmed |
| 00D5 | 213 | Current screen line length |
| 00D6 | 214 | Row where curosr lives |
| 00D7 | 215 | Last inkey/checksum/buffer |
| 00D8 | 216 | # of INSERTs outstanding |
| 00D9-00F0 | 217-240 | Screen line link table |
| 00F1 | 241 | Dummy screen link |
| 00F2 | 242 | Screen row marker |
| 00F3-00F4 | 243-244 | Screen color pointer |
| 00F5-00F6 | 245-246 | Keyboard pointer |
| 00F7-00F8 | 247-248 | RS-232 Rcv pntr |
| 00F9-00FA | 249-250 | RS-232 Tx pntr |
| 00FF-010A | 256-266 | Floating to ASCII work area |
| 0100-103E | 256-318 | Tape error log |
| 0100-01FF | 256-511 | Processor stack area |
| 0200-0258 | 512-600 | Basic input buffer |
| 0259-0262 | 601-610 | Logical file table |
| 0263-026C | 611-620 | Device # table |
| 026D-0276 | 621-630 | Sec Adds table |
| 0277-0280 | 631-640 | Keybd buffer |
| 0285 | 645 | Serial bus timeout flag |
| 0286 | 646 | Current color code |
| 0287 | 647 | Color under cursor |
| 0288 | 648 | Screen memory page |
| 0289 | 649 | Max size of keybd buffer |
| 028A | 650 | Repeat all keys |
| 028B | 651 | Repeat speed counter |
| 028C | 652 | Repeat delay counter |
| 028D | 653 | Keyboard Shift/Control flag |
| 028E | 654 | Last shift pattern |
| 028F-0290 | 655-656 | Keyboard table settup pointer |
| 0291 | 657 | Keymode (Kattacanna) |
| 0292 | 658 | 0=scroll enable |
| 0293 | 659 | VIC chip control |
| 0294 | 660 | VIC chip command |
| 0295-0296 | 661-662 | Bit timing |
| 0297 | 663 | RS-232 status |
| 0298 | 664 | # bits to send |
| 0299-029A | 665 | RS-232 speed/code. |
| 029B | 667 | RS232 receive pointer |
| 029C | 668 | RS232 input pointer |
| 029D | 669 | RS232 transmit pointer |
| 029E | 670 | RS232 output pointer |
| 029F-02A0 | 671-672 | IRQ save during tape I/O |
| 0300-0301 | 768-769 | Error message link |
| 0302-0303 | 770-771 | Basic warm start link |
| 0304-0305 | 772-773 | Crunch Basic tokens link |
| 0306-0307 | 774-775 | Print tokens link |
| 0308-0309 | 776-777 | Start new Basic code link |
| 030A-030B | 778-779 | Get arithmetic element link |
| 0314-0315 | 788-789 | Hardware interrupt vector (EABF) |
| 0316-0317 | 790-791 | Break interrupt vector (FED2) |
| 0318-0319 | 792-793 | NMI interrupt vector (FEAD) |
| 031A-031B | 794-795 | OPEN vector (F40A) |
| 031C-031D | 796-797 | CLOSE vector (F34A) |
| 031E-031F | 798-799 | Set-input vector (F2C7) |
| 0320-0321 | 800-801 | Set-output vector (F309) |
| 0322-0323 | 802-803 | Restore I/O vector (F3F3) |
| 0324-0325 | 804-805 | INPUT vector (F20E) |
| 0326-0327 | 806-807 | Output vector (F27A) |
| 0328-0329 | 808-809 | Test-STOP vector (F770) |
| 032A-032B | 810-811 | GET vector (F1F5) |
| 032C-032D | 812-813 | Abort I/O vector (F3EF) |
| 032E-032F | 814-815 | USR vector (FED2) |
| 0330-0331 | 816-817 | LOAD link |
| 0332-0333 | 818-819 | SAVE link |
| 033C-03FB | 828-1019 | Cassette buffer |
| 0400-0FFF | 1024-4095 | 3K RAM expansion area |
| 1000-1FFF | 4096-8191 | Normal Basic memory |
| 2000-7FFF | 8192-32767 | Memory expansion area |
| 8000-8FFF | 32768-36863 | Character bit maps |
| 9000-900F | 36864-36879 | Video Interface Chip |
| 9110-912F | 37136-37167 | 6522 Interface Chips |
| 9400-95FF | 37888-38399 | Alternate Colour Nybble area |
| 9600-97FF | 38400-38911 | Main Colour Nybble area |
| A000-BFFF | 40960-49151 | Plug-in ROM area |
| C000-FFFF | 49152-65535 | ROM: Basic and Operating System |

# VIC-20

# EL ORDENADOR PERSONAL AMPLIABLE CON COLOR Y SONIDO.



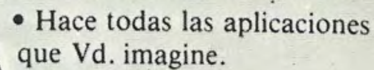

**49.500 Ptas.**
**COLOR-SONIDO**

*Así es el VIC-20*

- Lenguaje BASIC extendido.
- Sistema operativo COMMODORE.
- 5 K RAM ampliable a 32 K.
- 16 colores, 4 generadores de sonido.
- 66 caracteres gráficos.
- Periféricos disponibles:
  - Cassette.
  - Impresora de agujas.
  - Unidad de disco de 170 K.

*Así hace las cosas el VIC-20*

- Enseña informática.
- Efectúa todo tipo de cálculos matemáticos.
- Realiza funciones docentes.
- Se encarga de múltiples tareas profesionales.
- Proporciona divertidos momentos de ocio.
- Ayuda a planificar labores domésticas.
- Hace todas las aplicaciones que Vd. imagine.





## GRATIS

Con la adquisición de su VIC-20 recibirá además:

- MANUAL DEL USUARIO.
- INTRODUCCION AL LENGUAJE DE PROGRAMACION BASIC.
- Y 17 PROGRAMAS DE PRACTICAS (en dos cassettes).




**commodore COMPUTER**

Distribuidor exclusivo para España:

Microelectrónica y Control, S.A.
Taquigrafo Serra, 7 5.°. Barcelona-29
Princesa, 47 3.° G. Madrid-8

***De venta en tiendas especializadas.***

SLOGAN